Being on-time is not easy…

Virtual Platforms for Mixed Time-Criticality Applications: The CompSOC Platform and SDF3 Design Flow

Systems-on-Chip (SoC) complexity increases as a growing number of applications are integrated and executed on contemporary systems. These applications consist of communicating tasks mapped on heterogeneous multi-processor platforms with distributed memory hierarchies that strike a good balance between performance, cost, power consumption and flexibility. Complexity is further increased by an increasing number of *use-cases*, which are different combinations of concurrently running applications. The applications have *mixed time-criticality*, which is a mix between firm, soft, and no *real-time requirements*. Firm real-time requirements must always be satisfied to prevent unacceptable output quality loss, while occasional failures to meet soft requirements can be tolerated. Lastly, non-real-time applications do not have well defined timing requirements, but must still be responsive.

Different applications are often developed by different companies using different design flows and verification methods. Traditionally, only when applications are integrated in the same platform is the system as a whole verified. Since applications interfere with another, latent bugs that were not found by testing the applications in isolation, are often triggered. As a result, *debugging* to get the system to work often takes months, which costs a lot of money. The interference between applications arises from the need to reduce cost: platform resources, such as processors, interconnect, and memories, are *shared between applications*.

The **CompSOC platform** (www.compsoc.eu) addresses these problems by executing each application in an independent *virtual platform*, and by using the **SDF3 design flow** (www.es.ele.tue.nl/sdf3) that automatically analyses firm real-time applications and maps them on a virtual platform. The CoMPSoC virtualization technology relies on two complexity-reducing concepts: *composability* and *predictability*, detailed as follows. Composable virtual platforms are completely isolated and cannot affect each other's temporal behaviors by even a single clock cycle. They are hence *virtualized in terms of actual execution time*, enabling applications to be designed, developed and verified in isolation.

The virtual platforms are also predictable, which means that all platform and application interference is bounded. This makes them *virtualized in terms of performance bounds*, such as upper bounds on latency or lower bounds on throughput. This enables firm real-time applications to be verified using formal performance analysis frameworks, such as dataflow analysis.

The CompSOC platform & SDF3 design flow have many facets, all of which are research topics in their own right:

- **Hardware**: multi-processor, network on chip, DRAM memory controller – see Figure 2.
- **Software**: microkernel, RTOS, resource management middleware, etc.
- With: **virtualization**, **real-time power management**, mixed criticality.
- **Applications**: adaptive embedded video decoders, audio, games, control, etc.
- **Demonstrators**: multiple FPGAs connected wirelessly and wired (e.g. automotive CAN) – see Figure 2.
- **Formal performance analysis**: advanced scenario-aware dataflow theory, modeling platforms (Figure 3).
- **Design flow**: automatically generate CompSOC platforms using SDF3 dataflow tools.

Curious? The CompSOC platform is used in the Embedded Systems Lab. Master course (5KK03). Bachelor end projects, Master internships and project are available on all the above topics. Come and join the CompSOC and the SDF3 team in the Electronic Systems group!

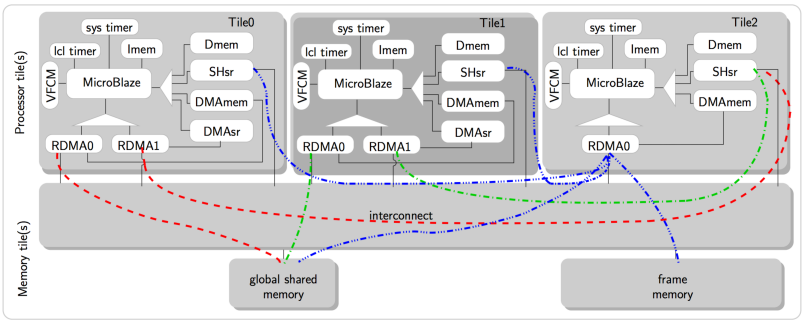

Figure 1 Example CompSOC architecture, consisting of three processor tiles, network on chip, and DRAM.

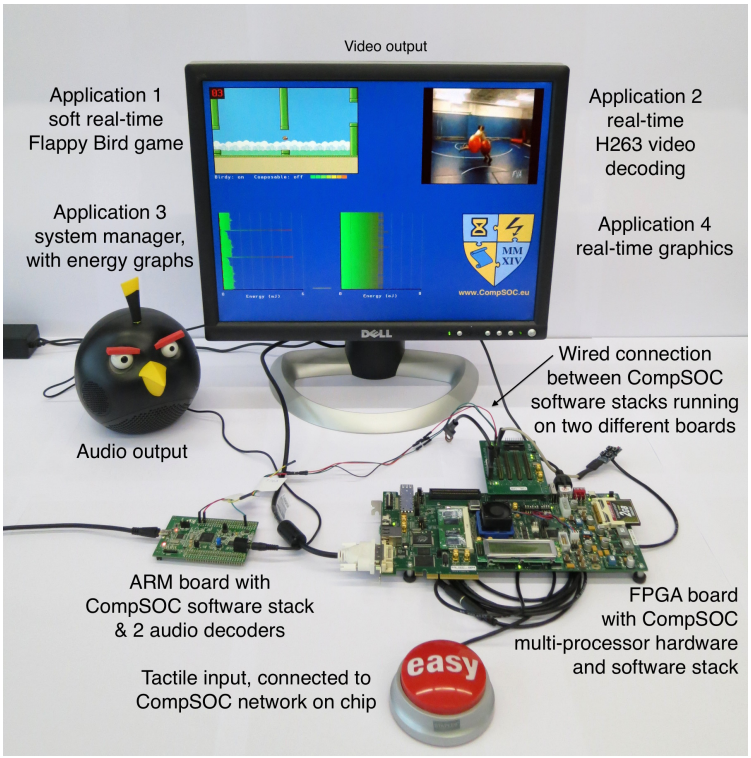

Figure 2 - FPGA board (right) with CompSOC hardware & software running four applications, communicating with ARM board running CompSOC software (left) running two applications, and output sound (Angry Bird) & video (screen).

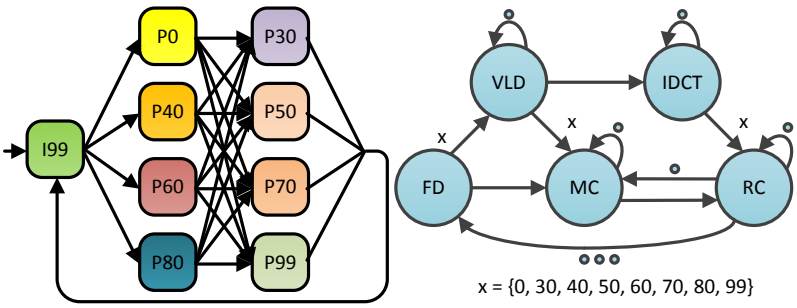

Figure 3 - Example application graph and dataflow model for performance analysis with SDF3.